

Sidebar

Programming · VS Code Language Settings Cheat Sheet

Settings that belong to one language or tool: per-language overrides, Python (extension and Pylance), JavaScript/TypeScript, Markdown, Prettier and ESLint. Exact ID, what it does, and its default. Sheet 6 in the VS Code series; see the tip below about which versions were checked.

PER-LANGUAGE OVERRIDES

- `"[python]": { ... }` In `settings.json`: any setting placed inside applies to that language only
- `"[javascript]": { "editor.tabSize": 2 }` Example: two-space indents for JavaScript, whatever the global setting
- `"[markdown]": { "editor.wordWrap": "on" }` Example: wrap lines in Markdown only
- `editor.defaultFormatter` Set inside a language block to choose its formatter, e.g. `"esbenp.prettier-vscode"`
- `@lang:python` Settings-editor filter that lists only settings you can override per language
- `files.associations` Treat file patterns as a language, e.g. `{ "*.cmd": "bat" }`
- `Ctrl+K M` Change Language Mode for the current file (also: click the language in the status bar)

PYTHON TESTING

- `python.testing.pytestEnabled` Use pytest for the Testing view (default `false`)
- `python.testing.unittestEnabled` Use unittest instead (default `false`)
- `python.testing.pytestArgs` Extra pytest arguments, e.g. `["tests"]` (default `[]`)
- `python.testing.unittestArgs` unittest arguments (default `-v -s . -p *test*.py`)
- `python.testing.autoTestDiscoverOnSaveEnabled` Rediscover tests when you save (default `true`)
- `python.testing.cwd` Working folder for test runs (default none)

PYTHON EXTENSION: BASICS

- `python.defaultInterpreterPath` Interpreter used when none is selected (default `python`)
- `Python: Select Interpreter` Command Palette command: choose the interpreter or virtual environment for this workspace
- `python.terminal.activateEnvironment` Activate the selected environment in new terminal (default `true`)
- `python.terminal.activateEnvInCurrentTerminal` Also activate it in the terminal that is already open (default `false`)
- `python.envFile` Path to a `.env` file loaded for runs (default `${workspaceFolder}/.env`)
- `python.createEnvironment.trigger` `prompt` (default) or `off`: offer to create a venv for projects
- `python.venvPath` Folder that contains your virtual environments (default empty)
- `python.languageServer` `Default` (default), `Jedi`, `Pylance`, `None`

PYLANCE: CHECKING & IMPORTS

- `python.analysis.typeCheckingMode` `off` (default), `basic`, `standard`, `strict`: how type problems are reported
- `python.analysis.diagnosticMode` `openFilesOnly` (default) or `workspace`: which files are analysed
- `python.analysis.autoImportCompletions` Offer symbols from unimported modules, with import (default `false`)
- `python.analysis.extraPaths` Extra folders searched for modules (default `[]`)
- `python.analysis.autoSearchPaths` Search `src` folders automatically (default `true`)
- `python.analysis.importFormat` `absolute` (default) or `relative` for generated imports
- `python.analysis.indexing` Index installed packages for faster completions (default `true`)
- `python.analysis.exclude / .include` Folders left out of / limited to analysis (default `[]`)
- `python.analysis.diagnosticSeverityOverrides` Change a rule's level, e.g. `{ "reportMissingImports": "warn" }`

PYLANCE: HINTS & EXTRAS

<code>python.analysis.inlayHints.variableTypes</code>	Show inferred variable types inline (default <code>false</code>)
<code>python.analysis.inlayHints.functionReturnTypes</code>	Show inferred return types inline (default <code>false</code>)
<code>python.analysis.inlayHints.callArgumentNames</code>	<code>off</code> (default), <code>partial</code> , <code>all</code> : show parameter names at call sites
<code>python.analysis.inlayHints.pytestParameters</code>	Show types for pytest fixtures (default <code>false</code>)
<code>python.analysis.completeFunctionParens</code>	Add <code>()</code> when you accept a function completion (default <code>false</code>)
<code>python.analysis.autoFormatStrings</code>	Typing <code>{ }</code> in a string adds the <code>f</code> prefix (default <code>false</code>)
<code>python.analysis.stubPath</code>	Folder for your own type stubs (default <code>typings</code>)
<code>python.analysis.useLibraryCodeForTypes</code>	Infer types from library source when there are no stubs (default <code>true</code>)

MARKDOWN (BUILT IN)

<code>Ctrl+Shift+V / Ctrl+K V</code>	Open the Markdown preview / open it beside the editor
<code>markdown.preview.breaks</code>	Single line breaks become line breaks in the preview (default <code>false</code>)
<code>markdown.preview.fontSize / .lineHeight</code>	Preview text size and spacing (defaults 14 and 1.6)
<code>markdown.preview.scrollPreviewWithEditor</code>	Preview follows the editor's scrolling (default <code>true</code>)
<code>markdown.preview.scrollEditorWithPreview</code>	Editor follows the preview's scrolling (default <code>true</code>)
<code>markdown.preview.openMarkdownLinks</code>	<code>inPreview</code> (default) or <code>inEditor</code>
<code>markdown.validate.enabled</code>	Check links and headings for errors (default <code>false</code>)
<code>markdown.updateLinksOnFileMove.enabled</code>	<code>never</code> (default), <code>prompt</code> , <code>always</code> : fix links when files move
<code>markdown.editor.drop.enabled / .filePaste.enabled</code>	<code>smart</code> (default), <code>always</code> , <code>never</code> : insert a link when you drop or paste a file
<code>markdown.styles</code>	Your own CSS files for the preview (default none)

JAVASCRIPT & TYPESCRIPT (BUILT IN)

<code>javascript.* / typescript.*</code>	Most settings come in twins; the examples below show one pair
<code>typescript.preferences.quoteStyle</code>	Quotes in generated code: <code>auto</code> (default), <code>single</code> , <code>double</code>
<code>typescript.preferences.importModuleSpecifier</code>	<code>shortest</code> (default), <code>relative</code> , <code>non-relative</code> , <code>project-relative</code>
<code>typescript.updateImportsOnFileMove.enabled</code>	Update imports when you move a file: (default), <code>always</code> , <code>never</code>
<code>typescript.suggest.autoImports</code>	Auto-import suggestions (default <code>true</code>)
<code>typescript.format.semicolons</code>	<code>ignore</code> (default), <code>insert</code> , <code>remove</code>
<code>typescript.inlayHints.parameterNames.enabled</code>	<code>none</code> (default), <code>literals</code> , <code>all</code>
<code>typescript.inlayHints.variableTypes.enabled</code>	Show variable types inline (default <code>false</code>)
<code>js/ts.implicitProjectConfig.checkJs</code>	Type-check plain <code>.js</code> files (default <code>false</code>)
<code>typescript.tsserver.maxTsServerMemory</code>	Memory limit for the TypeScript server, in MB (3072)
<code>typescript.tsdk</code>	Path to a TypeScript <code>lib</code> folder to use instead of the built-in one (default)

PRETTIER EXTENSION (ESBENP.PRETTIER-VSCODE)

<code>prettier.requireConfig</code>	Format only when a Prettier config file exists (default <code>false</code>)
<code>prettier.configPath</code>	Explicit path to a config file (default none)
<code>prettier.ignorePath</code>	File listing paths not to format (default <code>.prettierignore</code>)
<code>prettier.useEditorConfig</code>	Read <code>.editorconfig</code> (default <code>true</code>)
<code>prettier.printWidth</code>	Line length before wrapping (default 80)
<code>prettier.tabWidth / .useTabs</code>	Indent width (default 2) and tabs instead of spaces (default <code>false</code>)
<code>prettier.singleQuote / .semi</code>	Single quotes (default <code>false</code>) / semicolons (default <code>true</code>)
<code>prettier.trailingComma</code>	<code>all</code> (default), <code>es5</code> , <code>none</code>

ESLINT EXTENSION (DBAEUMER.VSCODE-ESLINT)

- `eslint.enable` Turn the extension on or off (default `true`)
- `eslint.run` Lint `onType` (default) or `onSave`
- `eslint.validate` Language IDs to lint; empty (default) lets the extension detect them
- `eslint.format.enable` Use ESLint as a formatter (default `false`)
- `eslint.codeActionsOnSave.mode` `all` (default) or `problems`; what a save-time fix may change
- `eslint.useFlatConfig` Force flat-config mode on or off; empty (default) lets ESLint decide
- `eslint.workingDirectories` Folders ESLint treats as project roots, useful in monorepos
- `eslint.lintTask.enable` Add a task that lints the whole workspace (default `false`)

A REASONABLE STARTER SET (MY SUGGESTIONS, NOT DEFAULTS)

- `"python.analysis.typeCheckingMode": "basic"` A useful middle ground; `off` is the default
- `"python.analysis.autoImportCompletions": true` Auto-import suggestions in Python
- `"python.terminal.activateEnvInCurrentTerminal": true` Activates the venv in an already open terminal
- `"[javascript]": { "editor.tabSize": 2 }` The usual two-space convention for web code
- `"[markdown]": { "editor.wordWrap": "on" }` Wrap prose files, not code
- `"markdown.validate.enabled": true` Flags broken Markdown links as you write
- `"typescript.updateImportsOnFileMove.enabled": "always"` No prompt when you move files

EXTENSION SETTINGS CHANGE WITH THE EXTENSION'S VERSION

JavaScript/TypeScript and Markdown settings are built into VS Code, but the Python, Pylance, Prettier and ESLint settings come from extensions and can change when those update. This sheet was checked against Python 2026.6.0, Pylance 2026.4.1, Prettier 12.4.0 and ESLint 3.0.34. Search the exact ID in the Settings editor to confirm it still exists in yours.

PRETTIER IGNORING YOUR VS CODE SETTINGS? LOOK FOR A CONFIG FILE

When a `.prettierrc` (or other Prettier config) or an `.editorconfig` exists in the project, it takes priority over the `prettier.*` settings in VS Code. Remove or rename the config file if you want VS Code's settings to be used.